

PŻ

---

DJANGO 03

# KONTYNUJEMY SZABLONY

---

```
def spis_pacjentow(request):
    wszyscy = Pacjent.objects.all()
    return render(request, template_name: 'pacjenci.html', context: {'imiona_pacjentów': wszyscy}) #wyświetla
```

Przypomnienie:

Pobieramy modele z bazy danych, po czym

Przekazujemy do szablonu. W szablonie

Definiujemy co dalej

# KONTYNUUJEMY SZABLONY

```
1  <!doctype html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport"
6          content="width=device-width, user-scalable=no, initial-scale=1.0,
7          maximum-scale=1.0, minimum-scale=1.0">
8      <meta http-equiv="X-UA-Compatible" content="ie=edge">
9      <title>Document</title>
10 </head>
11 <body>
12     <h2>To beda moje filmy</h2>
13
14     {% for film in filmy %}<br>
15         <p>{{ film.tytul }}</p>
16     {% endfor %}
17
18 </body>
</html>
```

```
{% for imiona_pacjentów in imiona_pacjentów%}
<p>{{ imiona_pacjentów }}</p>
{% endfor %}
#dodane dynamiczne przekazywanie pojedynczych danych - taka niewielka pętla
</body>
</html>
```

Skoro w objecy było all  
to jest dostęp do wszystkiego.  
Po kropce można wybierać co  
wyświetlamy

# WARTO WSPOMNIEĆ O ORM

- ▶ Coś, co ułatwia nam, język, który tłumaczy kod na język bazy danych.

```
File "<console>", line 1, in <module>
NameError: name 'Pacjent' is not defined
>>>
>>> from pacjenci.models import Pacjent
>>> wszyscy = Pacjent.objects.all()
>>> wszyscy
<QuerySet [<Pacjent: Łukasz Piątek>, <Pacjent: Jan Kowalski>, <Pacjent: Ola Żarnecka>, <Pacjent: Staszek Kowalski>, <Pacjent: UW PU>]>
>>> |
```

# WARTO WYKORZYSTYWAĆ SHELL DO PRACY

- ▶ Mowa o konsoli pythona. Jeżeli nasze IDE nie ma tej opcji - to wpisujemy w terminalu python3 manage.py shell i tyle

```
>>> wszyscy
<QuerySet [<Pacjent: Łukasz Piątek>, <Pacjent: Jan Kowalski>, <Pacjent: Ola Żarnecka>, <Pacjent: Staszek Kowalski>, <Pacjent: UW PU>]>
>>> wszyscy[1]
<Pacjent: Jan Kowalski>
>>> wszyscy[1].Imię
'Jan'
>>> █
```

# WARTO WYKORZYSTYWAĆ SHELL DO PRACY

- ▶ Możemy przeglądać, ale też i zarządzać obiektami bazy danych. Można szukać.
- ▶ A na przykład zamiast poniższego get, dać filter, zrobić np komendę len(wszyscy) - policzyć

```
<QuerySet [<>Pacjent: Łukasz Piątek>, <Pacjent: Jan Kowalski>, <Pacjent: Ola Zarnecka>, <Pacjent: S
<Pacjent: UW PU]>
>>> wszyscy[1]
<Pacjent: Jan Kowalski>
>>> wszyscy[1].Imię
'Jan'
>>> jeden = Pacjent.object
```

Menedżer, który pozwala na przeglądanie ale też zarządzanie bazą danych

```
>>> jeden = Pacjent.objects.get(Imię='Jan')
>>> jeden
<Pacjent: Jan Kowalski>
>>>
```

# WARTO WYKORZYSTYWAĆ SHELL DO PRACY

- ▶ A może dodać obiekt do bazy danych:

```
>>> Pacjent.objects.create(Imię="Kasia", Nazwisko="Jakas")
```

```
<Pacjent: Kasia Jakas>
```

```
>>>
```

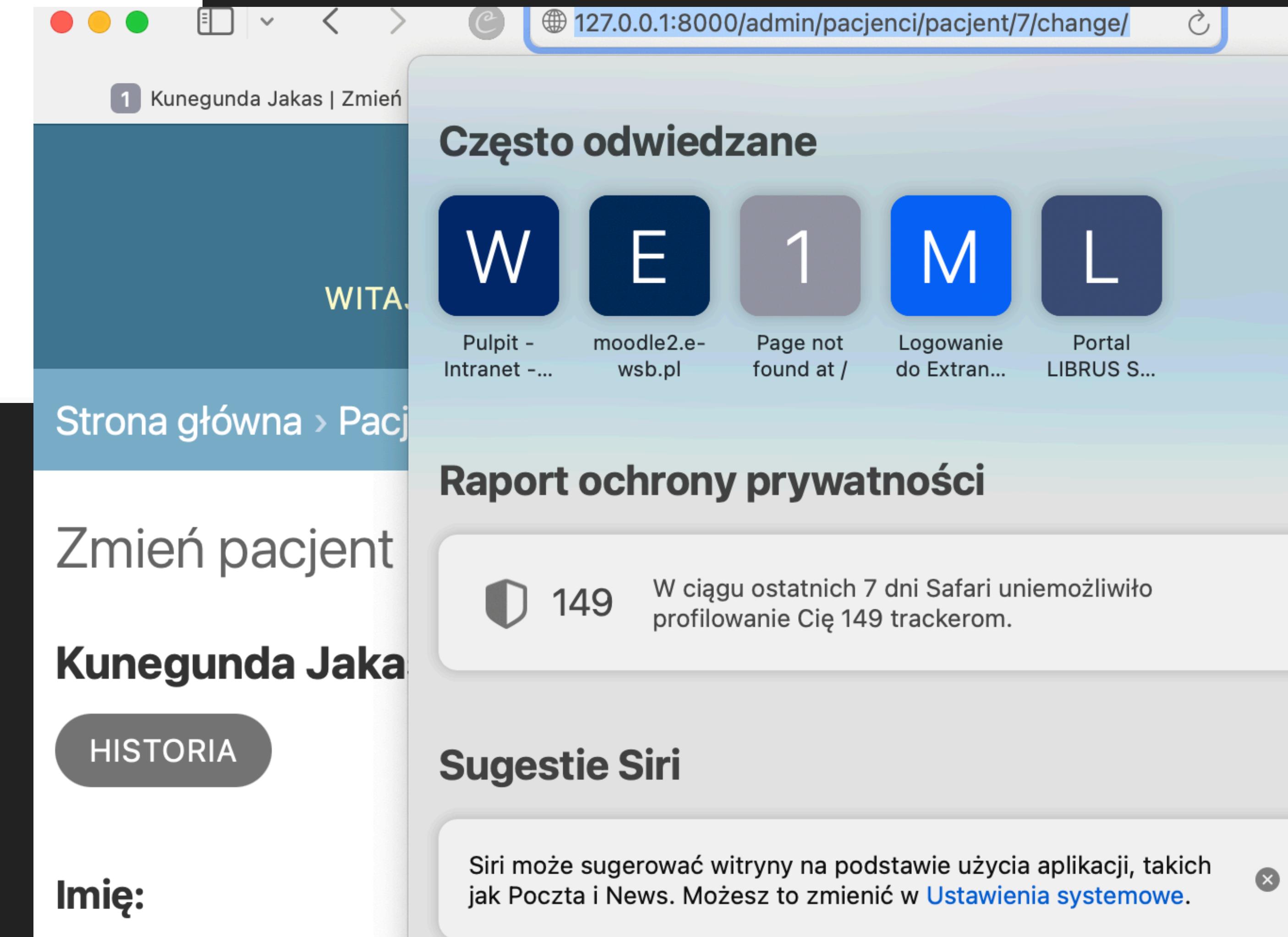
| <input type="checkbox"/> | IMIĘ    | NAZWISKO | DATA                 |
|--------------------------|---------|----------|----------------------|
| <input type="checkbox"/> | Kasia   | Jakas    | -                    |
| <input type="checkbox"/> | UW      | PU       | 23 listopada 2023    |
| <input type="checkbox"/> | dd      | dd       | 9 sierpnia 2023      |
| <input type="checkbox"/> | Staszek | Kowalski | 20 listopada 2023    |
| <input type="checkbox"/> | Ola     | Żarnecka | 10 października 1981 |
| <input type="checkbox"/> | Jan     | Kowalski | 10 października 1981 |
| <input type="checkbox"/> | Łukasz  | Piątek   | 16 listopada 2023    |

7 pacjents

# WARTO WYKORZYSTYWAĆ SHELL DO PRACY - TO CAŁY CZAS ORM

- W shellu można też dość szybko wykonywać aktualizacje. Zamiast Save - delete

```
>>> dwa = Pacjent.objects.get(id=7)  
>>> dwa  
<Pacjent: Kasia Jakas>  
>>> dwa.Imię ="Kunegunda"  
>>> dwa.save()  
>>>
```



# WSZYSTKO CO ROBILIŚMY W SHELLU, MOŻNA STOSOWAĆ W METODACH

---

```
def spis_pacjentow(request):
    wszyscy = Pacjent.objects.all()
    return render(request, template_name: 'pacjenci.html', cor
```

# WRACAMY DO SZABLOŃÓW

!!!!

```
<h2>Znajdować się tutaj będzie baza pacjentów</h2>
{{ ~ }}
```

```
{% for imiona_pacjentów in imiona_pacjentów%}
<p>{{ imiona_pacjentów }}</p>
{% endfor %}
#dodane dynamiczne przekazywanie pojedynczych danych
</body>
</html>
```

# WRACAMY DO SZABLOŃÓW - WAŻNA DOKUMENTACJA - SĄ WSZYSTKIE SKŁADNIKI

The screenshot shows a web browser window with the URL [docs.djangoproject.com/pl/4.2/topics/templates/](https://docs.djangoproject.com/pl/4.2/topics/templates/). The page content is as follows:

## Variables

A variable outputs a value from the context, which is a dict-like object mapping keys to values.

Variables are surrounded by `{{` and `}}` like this:

```
My first name is {{ first_name }}. My last name is {{ last_name }}.
```

With a context of `{'first_name': 'John', 'last_name': 'Doe'}`, this template renders to:

```
My first name is John. My last name is Doe.
```

Dictionary lookup, attribute lookup and list-index lookups are implemented with a dot notation:

```
{{ my_dict.key }}  
{{ my_object.attribute }}  
{{ my_list.0 }}
```

If a variable resolves to a callable, the template system will call it with no arguments and use its result instead of the callable.

---

**Tagi**

# WRACAMY DO SZABLOŃÓW - WAŻNA DOKUMENTACJA - SĄ WSZYSTKIE SKŁADNIKI

- W samym szablonie może jeszcze stosować wiele dodatków i filtrów. Przykład:

```
<body>
<h2>Ilość pacjentów w klinice: {{ imiona_pacjentów|length }}</h2>
<p><b>ten krok mi nie zadziałał...</b></p>
|
```

```
L2  {% if imiona_pacjentów|length == 7 %}
L3  <h2> Działa jak spełniony</h2>
L4  {% endif %}
L5  <h2>Ilość pacjentów w klinice: {{ imiona_pacjentów|length }}</h2>
L6
```

# WRACAMY DO SZABLOŃÓW - WAŻNA DOKUMENTACJA - SĄ WSZYSTKIE SKŁADNIKI

---

- ▶ Połączenie HTMLa i kodu Python dane coraz to ciekawsze efekty

```
<h2>Ilość pacjentów w klinice: {{ imiona_pacjentów|length }}</h2>
{% if imiona_pacjentów|length == 0 %}
<h2> Działa jak spełniony</h2>
{% else %}
    {% for imiona_pacjentów in imiona_pacjentów%}
        <p>{{ imiona_pacjentów }}</p>
    {% endfor %}
{% endif %}
```

# WRACAMY DO SZABLOŃÓW - WAŻNA DOKUMENTACJA - SĄ WSZYSTKIE SKŁADNIKI

- ▶ Połączenie HTMLa i kodu Python dane coraz to ciekawsze efekty

```
def spis_pacjentow(request):  
    wszyscy = Pacjent.objects.all() #można dodać tu count  
    #ilosc =len(wszyscy)  
    #wszyscy=[] - tu jesy pusta tablica dla warunku  
    return render(request, template_name: 'pacjenci.html', context: {'imiona_pacjentów':
```

Ilość pacjentów w klinice: 0

Działa jak spełniony

# DOŁĄCZANIE DO PROJEKTU PLIKÓW STATYCZNYCH

---

Pliki statyczne w Django to pliki, które są niezbędne do wyświetlenia pełnej strony internetowej, takie jak grafiki, JavaScript lub CSS. Pliki te nie są generowane dynamicznie przez serwer, ale są przechowywane w określonych katalogach i serwowane bez zmian. Django ma wbudowany mechanizm do zarządzania plikami statycznymi

# DOŁĄCZANIE DO PROJEKTU PLIKÓW STATYCZNYCH - METODA PIERWSZA

```
admin.py  
apps.py  
models.py  
tests.py  
urls.py
```

119     STATIC\_URL = 'static/'  
121

W pierwszej metodzie upewniamy się, że katalog static jest dodany do ustawień.

Domyślnie jest URL, dobrze jeszcze dodać katalog z plikami. Sam katalog ręcznie należy później dodać

```
17  
18  
19  
20 STATIC_URL = 'static/'  
21 STATICFILES_DIRS = ['statyczne']  
22 # Default primary key field type
```

# DOŁĄCZANIE DO PROJEKTU PLIKÓW STATYCZNYCH - METODA PIERWSZA

---

Następnie tworzymy sobie jakiś plik statyczny, który załadujemy do szablonu.

Standardowo zaczyna się od css ;-) Sam plik style.css działa jak klasyczny plik CSS i nie trzeba nic kombinować. Ale uwaga. Podłączenie jest nieco bardziej zakręcone niż w przypadku samego i zwykłego html

```
1 1 load static %}
2 <!doctype html>
3 <html lang="pl">
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport"
7     content="width=device-width, user-scalable=no, initial-sca
8   <meta http-equiv="X-UA-Compatible" content="ie=edge">
9   <title>Document</title>
10  <link rel="stylesheet" href="{% static 'style.css' %}"_
```

METODA PIERWSZA POLEGA NA TYM, ŻE TO  
STATIC JEST GLOBALNE I MOŻNA JE DZIELIĆ  
POMIĘDZY POSZCZEGÓLNYMI APLIKACJAMI W  
CAŁYM PROJEKCIE

Metoda pierwsza

# DOŁĄCZANIE DO PROJEKTU PLIKÓW STATYCZNYCH - METODA DRUGA

W drugiej metodzie możemy stworzyć katalogi static dla każdej aplikacji w projekcie z osobna.

The screenshot shows a file explorer on the left and a code editor on the right. The file explorer displays a project structure:

- Project root: kurs\_udemy (~/PyCharmProjects/kurs\_udemy)
- Subfolders: kurs\_udemy, pacjenci, migrations, static
- File: nowy.css

The code editor shows a Python template file (pacjenci/templates/index.html) with the following content:

```
3 <html lang="pl">
4   <head>
5     <meta charset="UTF-8">
6     <meta name="viewport"
7       content="width=device-width, user-scalable=no, initial-s
8     <meta http-equiv="X-UA-Compatible" content="ie=edge">
9     <title>Document</title>
10    <link rel="stylesheet" href="{% static 'nowy.css' %}">
11  <head>
12  <body>
13    <h2>Ilość pacjentów w klinice: {{ imiona_pacjentow|length }}</h2>
```

A red box highlights the 'static' folder in the file explorer and the 'nowy.css' file in the code editor, indicating they are the files being referenced in the template.

# ZAMIAST PLIKÓW STATYCZNYCH MOŻNA PODŁĄCZAĆ PLIKI MULTIMEDIALNE

---

Tu też w settings w raze W trzeba dodać ścieżkę do katalogu z mediami

Oryginalnie:

MEDIA\_URL = ,media'

Oraz MEDIA\_ROOT - podstawowe miejsce na przechowywanie plików

Folder, który podamy w ROOT, trzeba oczywiście stworzyć w głównej strukturze projektu

Wcześniej w przykładach modeli było pole graficzne - tam robiłem upload to 'twarze' - i mi się zrobił katalog twarze. Teraz jak jest zdefiniowany katalog na media, to tam się to będzie robiło - coś na kształt porządku - później jeszcze trzeba popracować nad ich wyświetlaniem, bo sam serwer Django tego nie lubi. NIE ZALECA SIĘ UŻYWAĆ DJANGO DO WYSWIETLANIA PLIKÓW

# ZAMIAST PLIKÓW STATYCZNYCH MOŻNA PODŁĄCZAĆ PLIKI MULTIMEDIALNE

```
2 from django.contrib import admin
3 from django.urls import path, include
4 from django.conf import settings
5 from django.conf.urls.static import static
#from pacjenci.views import test_response
6
7 urlpatterns = [
8     path('admin/', admin.site.urls),
9     path('pacjenci/', include('pacjenci.urls')) #to prowadzi do urls w aplikacji
10    ]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
11
```

---

**NA SERWER ZALECA SIĘ ROZWIĄZANIA  
ZEWNĘTRZNE**

**Do serwowania plików**